Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jernej Plavčak

Prototip enostavnega učnega sistema

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Ljubljana 2015

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jernej Plavčak

Prototip enostavnega učnega sistema

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: viš. pred. dr. Igor Rožanc

Ljubljana 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomskem delu predstavite idejno rešitev ter sistematičen razvoj prototipa enostavnega učnega sistema, ki vsebuje bistvene funkcionalnosti tovrstnih sistemov: zmožnost organizacije različnih tečajev, učinkovito podajanje učne snovi, enostavno evidentiranje prisotnosti, uporabno preverjanje znanja s pomočjo kvizov ter nazorno evidenco ocen z analizo uspešnosti. To naj sistem implementira s smiselnim naborom akcij za tri vrste uporabnikov (skrbnika, učitelja in učenca). Poudarek pri izvedbi naj bo na enostavnosti uporabe ter domišljenih rešitvah za preverjanje znanja s kvizi.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Jernej Plavčak sem avtor diplomskega dela z naslovom:

Prototip enostavnega učnega sistema

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Igorja Rožanca,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 23. novembra 2015 Podpis avtorja:

Zahvaljujem se svojemu mentorju viš. pred. dr. Igorju Rožancu za vso podporo in vzpodbudo pri izdelavi diplomskega dela.
Zahvala pa gre tudi moji družini in punci, ki so me vzpodbujali skozi celotno študijsko leto in v času nastajanja diplomskega dela.

Diplomsko delo posvečam svoji družini in punci.

Kazalo

Slike

Seznam uporabljenih kratic

kratica	angleško	slovensko
DMX	Data Mining Extensions	Razširitev podatkovnega rudarjenja
GUI	Graphical User Interface	Grafični uporabniški vmesnik
GPL	General Public License	Splošna javna licenca
HTML	Hyper Text Markup Language	Jezik za označevanje nadbesedila
PHP	Hypertext Preprocessor	Programski jezik PHP
SQL	Structured Query Language	Poizvedovalni jezik
T-SQL	Transact-SQL	Transakcijsko poizvedovalni jezik
XML	Extensible Markup Language	Razširljiv označevalni jezik

Povzetek

V okviru diplomskega dela je bil razvit prototip enostavnega učnega sistema, ki je namenjen učiteljem in učencem, za uporabo pa skrbi skrbnik. Učiteljem ponuja sistem številne funkcionalnosti: sestavljanje učne snovi, evidenco udeležencev in ocen ter ocenjevanje učencev s pomočjo kvizov. Skrbnik vodi evidenco uporabnikov, dodaja nove predmete učiteljem in potrjuje uporabnike. Učenci lahko pridobivajo svoje znanje preko lekcij in ga utrjujejo s pomočjo kvizov.

Obstoječi sistemi za upravljanje znanja so ponavadi preobsežni in ponujajo ogromno (preveč) funkcionalnosti. Učitelji in druge uporabniške skupine, ob prvi uporabi ponavadi ne znajo primerno uporabljati sistema, zato potrebujejo dodatna usposabljanja. Na podlagi tega smo zasnovali preprost prototip učnega sistema, ki je enostaven za uporabo in zanj uporabniki ne potrebujejo predhodnega usposabljanja.

V prvem poglavju naredimo pregled obstoječih sistemov za upravljanje znanj. V naslednjem poglavju podrobno opišemo uporabljene tehnologije, zahteve za izdelavo aplikacije in funkcionalnosti uporabniških skupin. Glavni del diplomskega dela opisuje razvoj aplikacije. V tem delu je opisana implementacija podatkovne baze, dvonivojska arhitektura in grafični vmesniki posameznih uporabniških skupin (učenec, učitelj in skrbnik). Na koncu poglavja sta opisana testiranje in analiza aplikacije. Rezultat diplomskega dela je delujoč prototip enostavnega učnega sistema, ki ga je možno uporabiti kot osnovo za resnejši učni sistem.

Ključne besede: prototip, učni sistem, lekcija, kviz, evidenca ocen.

Abstract

In my diploma thesis, a prototype of a simple education system to be used by teachers and learners alike, and managed by an administrator has been developed. To teachers, the system offers many functionalities, such as the formation of learning materials, records of participants and grades, and learners assessments through quizzes. The administrator supervises user records, adds new subjects to teachers and confirms users. Learners can acquire knowledge through lessons while strengthening it with the help of quizzes.

Existing learning management systems are often too extensive and offer too many functionalities. When utilizing them for the first time, teachers and other user groups often do not possess the skill set to use the system properly, and therefore need additional training. Because of the reasons listed above, a prototype of a simple, easy to use education system has been designed, where the users need not undergo any additional training in order to grasp the systems functioning.

In the first chapter, a review of existing learning management systems is presented. In the next one, the technologies used, the requirements for developing the application, as well as functionalities of user groups are described in detail. The main part of the diploma thesis deals with the development of the application. In this part, the implementation of the database, two-tier architecture and graphical user interfaces for all user groups (teacher, learner and administrator) are described. At the end of the chapter, the testing and analysis of the application is delineated. The result of the diploma thesis is a working prototype of a simple education system that can be used as a

foundation for a more significant learning system.

Keywords: prototype, education system, lesson, quiz, gradebook.

Poglavje 1

Uvod

V današnjem času obstaja veliko različnih sistemov za upravljanje znanja [16], med katere prištevamo tudi odprtokodne in plačljive. Sistemi za upravljanje znanja so ponavadi zelo obsežni, ponujajo ogromno orodij in vtičnikov, ki jih je možno vgraditi. Obsežnost sistema povečuje potrebo po bolj temeljitem vzdrževanju, testiranju novih vtičnikov, popravljanju morebitnih sistemskih in varnostnih napak. Učitelji in druge uporabniške skupine, ki se prvič srečajo s tem sistemov, potrebujejo ponavadi dodatna usposabljanja. Uporabniki imajo na voljo veliko funkcionalnosti, ki jih ponavadi ne znajo uporabljati ali pa ne potrebujejo. Veliko izobraževalnih ustanov in posameznikov, ki poučujejo skupine ljudi, se ponavadi ne poslužuje teh sistemov ravno zaradi zgoraj naštetih razlogov.

Cilj diplomskega dela je izdelati prototip enostavnega učnega sistema, ki bo dovolj enostaven za uporabo in bo zajemal bistvene funkcionalnosti uporabniških skupin (učitelj, učenec in skrbnik). Nekaj bistvenih funkcionalnosti:

- skrbniku omogočimo potrjevanje uporabnikov,
- učitelju omogočimo sestavljanje in ocenjevanje kvizov, sestavljanje učne snovi in analizo rezultatov,
- učencu omogočimo vodeno učenje skozi reševanje kvizov.

To so bistvene funkcionalnosti, ki jih mora sistem zadovoljiti, še zdaleč pa niso zajete vse. V nadaljevanju sta opisani motivacija in ideja za izdelavo aplikacije.

1.1 Motivacija

Za izdelavo prototipa učnega sistema me je navdihnila družina Česnik, ki se ukvarja s kleklanjem in poučevanjem idrijske čipke [11], ki predstavlja pomemben del slovenske dediščine ter je uvrščena na Unescov seznam kulturnih dediščin. Ohranitev in poznavanje čipke je iz tega vidika še pomembnejša, česar se zavedajo tudi obiskovalci čipkarske šole v Idriji. Poleg pridobljenega praktičnega znanja je pomembno tudi poznavanje teorije, ki bi jo lahko učitelji preverjali s pomočjo sestavljenih kvizov.

1.2 Ideja

Prvotno se je ideja nanašala na izdelavo prototipa učnega sistema za mobilno aplikacijo. Ljudje s slabšim vidom bi pri učenju zaradi manjše velikosti zaslona imeli težave (npr. pri učenju kleklanja). Zaradi velikosti zaslona pametnih naprav, smo se v nadaljevanju odločili, da bomo izdelali namizno aplikacijo. Podkrepljena odločitev za izbiro namizne aplikacije je bila predvsem v poznavanju programskega jezika Java [12] in swingove knjižnjice.

Za prototip enostavnega učnega sistema želimo, da je dovolj enostaven za uporabo. Učitelji, učenci in skrbniki ob tem ne potrebujejo nobenega predhodnega usposabljana. Vso pomoč, ki jo potrebujejo pri uporabi lahko dobijo s pomočjo orodja za namige (primer uporabe prikazuje slika 4.14). Nalaganje učne snovi naj bo enostavno in zajema samo `pdf` datoteke. Kvizi so lahko sestavljeni tako, da pomagajo učencu utrditi pridobljeno znanje. Učitelj lahko na drugi strani brez sestavljanja pisnih izpitov pridobiva ocene učencev izključno iz kvizov.

Poglavje 2

Pregled obstoječih sistemov

Sistem za upravljanje znanja (angl. Learning Management System) [16] je program, ki je namenjen upravljanju in dostavi učnih vsebin ter virov učencem. Teh sistemov je zelo veliko, zato smo se odločili za tri in jih podrobneje pregledali. Izbrali smo naslednje: Moodle, Blackboard Learn in ATutor. Pri pregledu smo se osredotočili predvsem na uporabniške skupine in njihove funkcionalnosti.

2.1 Moodle

Moodle [15] je izobraževalni sistem, ki omogoča učenje preko spleta. Sistem je odprtokodni in zastonjski. Preveden je v več kot 120 jezikov. Fleksibilen spletni videz omogoča, da se sistem uporablja tudi na mobilnih napravah. Razvit je bil na podlagi pedagoških principov. Namen sistema je, da učiteljem omogoči ustvarjanje kvalitetnega učnega gradiva preko spleta. Poleg tega sistem ponuja prijetno okolje za sodelovanje med učenci in učitelji.

2.1.1 Opis sistema in funkcionalnosti

Začetna stran Moodle (na sliki 2.1) ponuja osnovne informacije o sistemu ter navodila za prijavo v sistem. Vsebino strani je možno poljubno preoblikovati. Obstaja več načinov registriranja uporabnikov:

- podatke za prijavo v sistem dobijo avtomatsko od drugega sistema,
- omogočeno jim je, da si naredijo lasten uporabniški račun,
- nove uporabnike ustvari skrbnik sistema in jim posreduje podatke za prijavo.

V sistemu ne moremo dostopati do nekaterih vsebin, če nimamo uporabniške vloge (npr. učitelj ali učenec). Gostom je na voljo vsebina tečajev (onemogočeno reševanje kvizov in oddaja nalog), če to omogočijo učitelji. Uporabnik, ki se prijavi v sistem, nima nobenih pravic, dokler ga ne potrdi skrbnik sistema.

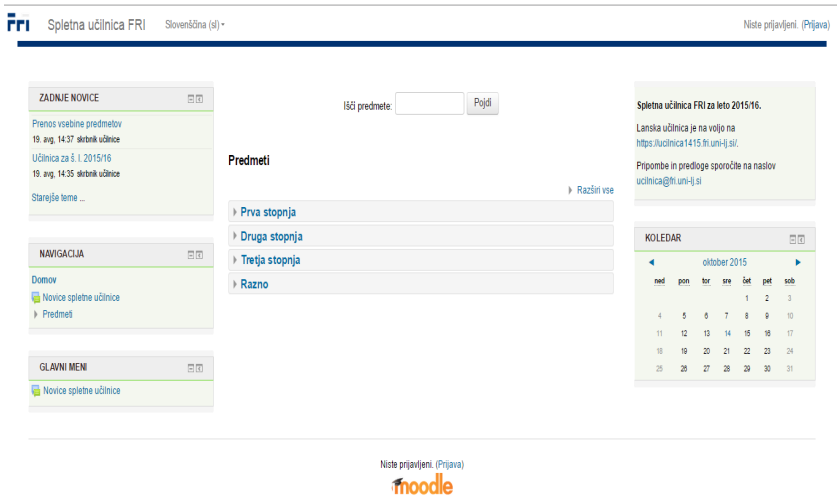
Osnovna struktura sistema temelji na tečajih. Tečaji predstavljajo strani znotraj sistema, s pomočjo katerih lahko učitelj predstavi svoje učno gradivo in določa aktivnosti (npr. kvize) za svoje učence. Učitelj lahko spreminja videz strani po potrebi. Obstajata dva načina vpisovanja v tečaj:

- učenec se lahko vpiše sam ali s pomočjo gesla, ki ga dobi za vpis v predmet,
- vpiše ga učitelj ali skrbnik sistema.

Sistem ponuja na voljo 4 osnovne funkcionalnosti:

- shranjevanje (slike, video posnetke in druge vrste datotek),
- ocenjevanje (npr. s pomočjo kvizov, anket in nalog),
- sodelovanje (npr. preko lekcij, forumov in Wiki),
- komuniciranje (npr. preko forumov, elektronske pošte in klepetalnice).

Skrbnik sistema lahko vidi, naredi in uredi vse strani (npr. doda določene funkcionalnosti uporabniku). Učitelj lahko prav tako v sklopu svojih tečajev naredi, vidi in uredi vse (npr. sestavlja kvize). Učitelj določa učencu kaj lahko vidi in ureja (npr. rešuje kvize) znotraj tečaja. Poleg uporabniških skupin (učitelj, skrbnik in učenec) je možno ustvariti tudi druge. Skrbnik lahko ustvari npr. vlogo starša in ji dodeli ustrezne funkcije.



Slika 2.1: Grafični prikaz spletne učilnice, ki uporablja Moodle.

2.1.2 Verzije Moodle

Osnovno ogrodje Moodle deluje na sistemih, ki podpirajo PHP. Sistem je na voljo v večih verzijah. Teh verzij je zelo veliko, saj se lahko v sistemu pojavljajo napake, ki jih je potrebno odpraviti. Novejše verzije vsebujejo popravke iz prejšnjih verzij, izgledajo lepše in ponujajo več funkcionalnosti. Nekaj značilnih verzij Moodle:

- Moodle 2.4,
- Moodle 2.5,
- Moodle 2.6,
- Moodle 2.7,
- Moodle 2.8,
- Moodle 2.9.

Glede na statistične podatke, ki jih ponuja spletna stran [21] je trenutno najbolj uporabljena verzija 2.9. Ena izmed vidnejših izboljšav te verzije je

lokacija vseh predmetnih ocen učenca v zavihku `Ocene` in sicer na njegovi osebni strani. S tem je prihranjen uporabnikov čas, ki ga potrebuje za ogled vseh ocen posazmeznih predmetov (ocena predmeta se nahaja na strani predmeta in ne na osebni strani). Verzija ima še veliko drugih izboljšav, katere so podrobno opisane na spletni strani [20].

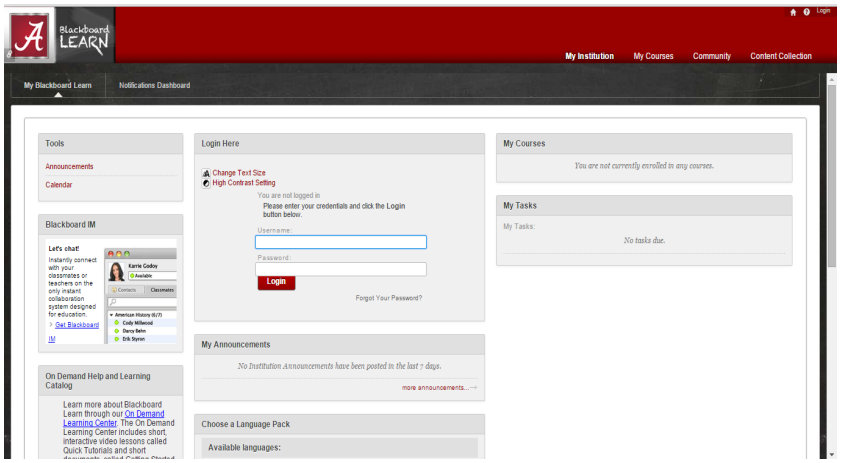
2.1.3 Vtičniki

Vtičnik je programska komponenta, ki doda funkcionalnosti programu. Na voljo je preko 1000 vtičnikov različnih vrst, ki se lahko vgradijo v sistem. Posamezni omogočajo preverjanje plagiatorstva, spreminjanje vizualnega izgleda tečajev, tvorjenje poročil in številne druge funkcionalnosti.

Uporaba vtičnikov zahteva ustrezno znanje, saj se pri nepravilni uporabi le-teh zlahka pojavijo napake, ki jih je potrebno odpraviti. Pomoč pri odpravljanju napak med drugim nudi tudi Moodlov forum [19]. Pri pregledu foruma smo ugotovili, da se veliko uporabnikov srečuje z napakami, ki jih sami stežka odpravijo. Uporabniki v večini primerov napake odpravijo s pomočjo skrbnika sistema ali nasvetov uporabnikov, ki imajo izkušnje ter so se že srečali s podobnimi napakami. Pri namestitvi dodatnih funkcionalnosti je potreben tehten premislek, da le-te ne predstavljajo dodatnega bremena uporabniku (npr. učiteljem). Prihaja pa tudi do sistemskih napak, ki jih ponavadi odpravijo razvijalci.

2.2 Blackboard Learn

Blackboard Learn (slika 2.2) [4] je virtualni sistem za učenje, ki ga je ustanovilo podjetje Blackboard Inc. Učiteljem omogoča upravljanje in urejanje tečajev, komunikacijo med učenci kot tudi samo ocenjevanje.



Slika 2.2: Grafični prikaz Blackboard Learn 9.1.

2.2.1 Komuniciranje in deljenje vsebine

Sistem ponuja bogata orodja za komunikacijo in izvajanje različnih diskusij med učenci in učitelji. Učencem omogoča seznanitev s potekom izvajanja predmeta in prestalimi obvestili. Učenci lahko medsebojno komunicirajo s pomočjo spletne klepetalnice in pošiljajo vprašanja po elektronski pošti učiteljem.

Učitelji na drugi strani objavljajo obvestila. Poleg tega znotraj svojega predmeta oziroma tečaja objavljajo še različne članke, naloge, video posnetke, itd. Ocenjevanje lahko poteka preko kvizov in testov. Učencu so tako vidne ocene kot tudi posamezni dogodki na koledarju (npr. kvizi ali testi), ki jih učitelj objavlja.

2.2.2 Blackboard Learn 9.1

Verzija Blackboard Learn 9.1 je trenutno najnovejša. Omogoča izboljšan sistem ocenjevanja, saj lahko presega 100 odstotkov. Učenec zlahka spreminja prikaz vrstnega reda ocen.

Sistem omogoča učitelju, da omeji dostop do testa preko filtriranja IP-naslovov. Na ta način lahko učenci rešujejo test samo na računalnikih, ki vsebujejo določen IP-naslov.

Orodje za skupine omogoča učiteljem boljši vpogled nad učenci, ki sodelujejo pri diskusijah. Učitelj lahko pooblasti učenca, da ustvari lastno skupino za diskusijo.

Kljub izboljšavam, ki jih nudi omenjen sistem pa se lahko pojavijo številne napake (npr. pri nalaganju več vrst datotek, pri imenih datotek, ki vsebujejo znake `#`, `%`, `&` ali presledek), ki so podrobno opisane na spletni strani [5]. Te napake so lahko velik problem, saj odvračajo izboraževalne ustanove ali druge uporabnike od uporabe takšnih sistemov. Pomembno je, da se napake čimprej odpravijo in da se pred integracijo dodatnih funkcionalnosti opravijo temeljita testiranja.

2.3 ATutor 2.2

ATutor 2.2 [3] je odrtokodni sistem za učenje, ki ponuja tečaje in učenje preko spleta. Preveden je v več kot 30 jezikov. Sistem je prilagojen za ljudi s posebnimi potrebami (za slepe in gluhoneme).

Poleg tega podpira integracijo različnih modulov. Eden izmed teh je `BigBlueButton`, ki ponuja sistem za upravljanje konferenc. S pomočjo tega modula lahko ustvarimo zvočne in video konference. Omogoča tudi dodajanja prezentacij `PowerPoint` in deljenje vsebine namizja z ostalimi uporabniki.

2.3.1 Uporabniške skupine in njihove funkcionalnosti

Sistem vsebuje naslednje uporabniške skupine:

- učenec,
- inštruktor,
- skrbnik.

V nadaljevanju so opisane glavne funkcionalnosti uporabniških skupin.

Učenec

Učenec ima na voljo naslednje funkcionalnosti:

- urejanje svojih osebnih podatkov (npr. slike),
- shranjevanje svojih virov,
- deljenje virov z ostalimi v skupini,
- oddajanje nalog vodji skupine ali inštruktorju,
- spreminjanje vizualnega izgleda svoje stran,
- komuniciranje z ostalimi udeleženci (preko klepetalnice, sporočil in forumov),
- reševanje testov in spremljanje svojih rezultatov,
- urejanje in dodajanje podatkov znotraj tečaja (npr. dodajanje povezav na spletne vire).

Inštruktor

Inštruktorj ima na voljo naslednje funkcionalnosti:

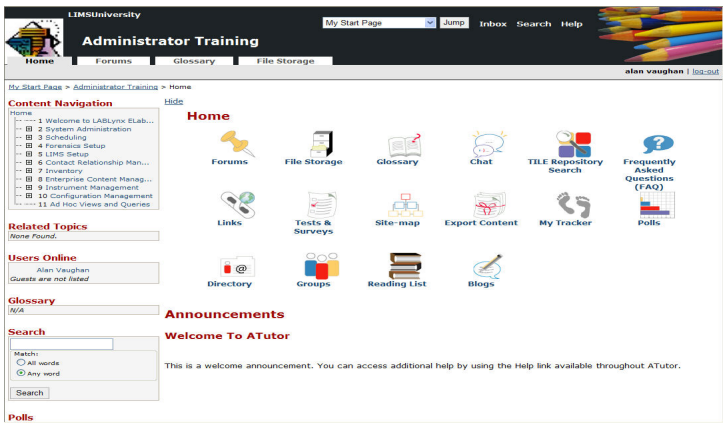
- ustvarjanje skupin,
- dodeljevanje učencev posameznim skupinam,
- omogočanje dostopanja gostov do privatnih tečajev preko spletnih povezav,
- pregledovanje rezultatov svojih učencev,
- shranjevanje učnih gradiv (te lahko delijo s skupinami ali posamezniki),
- sestavljanje učne vsebine (lahko uporablja HTML gradnike),
- dodeljevanje seznamov učnih gradiv,

- objavljanje obvestil in novic,
- sestavljanje testov različnih vrst (več možnih izbir, način resnično ali neresnično, določanje vrstnega reda, ...),
- ustvarjanje anket,
- dodeljevanje pravic učencem (npr. nekateri učenci lahko dodajajo povezave na spletne vire),
- komuniciranje z njimi preko sporočil, forumov, diskusij, ...

Skrbnik

Glavne funkcionalnosti, ki so na volju skrbniku (slika 2.3):

- nameščanje različnih modulov (lahko omogoča ali onemogoča),
- preverjanje identitete uporabnikov (inštruktorjev in učencev),
- dodeljevanje funkcionalnosti,
- nadzorovanje in vzdrževanje sistema,
- pregledovanje statističnih podatkov,
- upravljanje s sistemom za ustvarjanje kopij, uporabniki in tečaji.



Slika 2.3: Prikaz grafičnega vmesnika za skrbnike.

Sistem ponuja ogromno funkcionalnosti za uporabniške skupine. Prilagojen je za ljudi s posebnimi potrebami, kar predstavlja veliko prednost, saj večina izobraževalnih sistemov ni pripravljena na takšne uporabnike. Kljub temu pa ne ponuja vodenega učenja skozi reševanje kvizov, s pomočjo katerih bi učenci utrdili svoje znanje. Na ta način bi lahko učenci pridobivali ocene, učitelje pa bi razbremenili, saj jim nebi bilo potrebno sestavljati pisnih izpitov (na list papirja). Učenci lahko dodajajo povezave na svoje spletne vire znotraj tečaja. V primeru napačnih virov je potrebno vire odstraniti, kar predstavlja dodatno delo inštruktorjem. Ob morebitni uporabi teh virov lahko pride do napačnega znanja.

Poglavje 3

Zahteve aplikacije in uporabljene tehnologije

To poglavje opisuje zahteve, ki smo si jih zastavili pred začetkom izdelave diplomskega dela in uporabljene tehnologije. V prvem delu bomo opisali posamezne zahteve in odločitve za njih. Drugi del pa bo opisoval uporabljene tehnologije in programska okolja.

3.1 Zahteve za izdelavo namizne aplikacije

Vsak učni sistem potrebuje za svoje delovanje in prijetno uporabniško izkušnjo več uporabniških skupin. Vsaka upravlja svojo vlogo. Njihov namen je omejiti dostopnost do določenih vsebin aplikacije. Vsebina je vidna samo uporabnikom, ki pripadajo določeni skupini. Uporabniške skupine so naslednje: skrbnik, učitelj in učenec. Funkcionalnosti vsakega uporabnika so opisane v podpodpoglavjih, ki sledijo.

Zaščitenost uporabniškega gesla je zelo pomembna, zato smo se odločili za kriptiranje gesla z uporabo algoritma MD5 in soli.

Pozabljeno geslo lahko uporabniku včasih predstavlja velik problem, zato imamo možnost, da uporabniku geslo pošljemo na njegovo elektronsko pošto.

3.1.1 Skrbnik

Funkcionalnosti, ki jih ima skrbnik so naslednje:

- spreminjanje svojih osebnih podatkov,
- pregledovanje in potrjevanje registriranih uporabnikov,
- dodajanje novih predmetov učiteljev (če se izučijo za nove).

Vloga skrbnika je zelo pomembna, saj ne more biti vsak uporabnik učitelj in ne vsak uporabnik učenec (če ne obiskuje predmeta učitelja). Skrbnik potrdi vse uporabnike za katere smatra, da so resnični (učitelj obvesti skrbnika o tem, da je učitelj in pošlje seznam svojih učencev preko elektronske pošte).

3.1.2 Učitelj

Učitelj ima na voljo naslednje funkcionalnosti:

- spreminjanje svojih osebnih podatkov,
- urejanje podatkov o svojih predmetih (ime predmeta, opis, obdobje izvajanja predmeta in obdobje vpisa v predmet),
- pregledovanje svojih učencev (ime, priimek, telefon, elektronski naslov, datum registriranja, ...),
- vpisovanje ali izpisovanje učenca iz predmeta,
- sestavljanje lekcij in kvizov,
- prenašanje posameznih obdobij, lekcij, kvizov in vprašanj v novo obdobje,
- spremljanje rezultatov kvizov,
- preverjanje odgovorov,
- pregledovanje uspešnosti posameznikov za vsako obdobje predmeta,

- pregledovanje grafov (prikazujejo uspešnost posameznih obdobij predmeta, lekcij, kvizov in ocen učencev),
- pridobivanje ocen na podlagi kvizov,
- izvažanje ocen (v formatu datoteke excel).

Učitelj ima pomembno vlogo, saj sestavlja učno snov in kvize, ki jih rešujejo učenci. Popolna svoboda pri sestavljanju kvizov omogoča učitelju, da resnično sestavi kvize, ki jih učenci potrebujejo, da utrdijo svoje znanje.

3.1.3 Učenec

Funkcionalnosti učenca so:

- urejanje svojih osebnih podatkov,
- pošiljanje prošenj za vpis v predmete, katerih rok za prijavo je zamudil,
- pridobivanje znanja preko lekcij in kvizov,
- reševanje kvizov,
- pregledovanje svojih ocen.

Učenci sodijo med uporabniško skupino, ki je najštevilčnejša. Vloga učenca v prototipu aplikacije je pridobivanje svojega znanja preko lekcij in kvizov. Učenec ima tudi vpogled do svojih ocen.

3.2 Uporabljene tehnologije

V tem podpoglavju bomo opisali tehnologije, ki smo jih uporabili pri izdelavi aplikacije. Zanje smo se odloči zaradi njihove preproste sintakse, bogate dokumentacije in poznavanja.

3.2.1 Java

Programski jezik Java [12] je prenosljiv in objektno usmerjen (uporaba razredov). Ime Java izhaja iz otoka na Indoneziji kjer so pridelali veliko kave. Razvil ga je James Gosling skupaj s sodelavci v podjetju Sun Microsystems.

Na začetku se je projekt imenoval Oak kar pomeni hrast. Projekt so začeli leta 1991, njegov namen pa je bil zamenjati programski jezik C++. Sintaksa jezika je podobna C in C++ programskemu jeziku. Za sintaktično podobnost se je odločil, da bi bil programski jezik hitro razpoznaven med sistemskimi in aplikacijskimi razvijalci.

Javanske aplikacije lahko izvajamo na javanskih virtualnih napravah, neodvisno od arhitekture naprave na kateri program izvajamo. Prva različica Java 1.0 je izšla leta 1996.

Programski jezik je eden izmed najpopularneših in se uporablja že na več kot 9 miljardah napravah. Velik vtis je naredil na razvijalce spletnih aplikacij, ki delujejo na način odjemalec-strežnik. Odtod izhaja tudi njena priljubljenost. Priljubljena knjižnjica za uporabo grafičnih uporabniških vmesnikov je Swing. Knjižnjica ponuja veliko sofisticiranih GUI komponent (tabele, drevesa, sezname, ...).

3.2.2 Microsoft SQL Server

Microsoft SQL Server (MSSQL) [28] je relacijski sistem za upravljanje s podatki. Razvilo ga je podjetje Microsoft. Zasnovan je kot podatkovni strežnik s primarno funkcijo shranjevanja in vračanja podatkov na zahtevo aplikacije. Napisan je bil s pomočjo programskega jezika C in C++ [6].

SQL ali povpraševalni jezik je najbolj razširjen in standardiziran povpraševalni jezik za delo s podatkovnimi zbirkami. Določa ga Ameriški državni inštitut za standarde in Mednarodna organizacija za standardizacijo.

Poizvedovalni jezik, ki ga uporablja MSSQL je T-SQL[29]. T-SQL je lastniška razširitev SQL podjetij Microsoft in Sybase. Razširitev omogoča proceduralno programiranje, uporabo lokalnih spremenljivk, različno podporo

funkcijam za obdelavo nizov, datumov, matematičnih izrazov itd. T-SQL je ključen za uporabo MSSQL. Vse aplikacije, ki komunicirajo preko primerka MSSQL, storijo to s pošiljanjem T-SQL.

3.3 Programska okolja

V podpodpoglavjih so opisana programska okolja, ki so nam olajšala izdelavo aplikacije. Za oblikovanje posameznih ikon in slik smo uporabili Adobe Photoshop in Adobe Illustrator. Izdelava podatkovne baze je potekala v SQL Server 2014 Management Studio. Razvoj grafičnega uporabniškega vmesnika pa v programu NetBeans IDE 8.0.2. Delo nam je zelo olajšal program Notepad++. Omogočil nam je hitro iskanje določenih nizov v programski kodi. Njegova učinkovita uporaba pa se je izkazala pri spreminjanju nizov v podobnih programskih funkcijah (npr. funkcije za dodajanje lekcij, kvizov in vprašanj) za hitrejše pisanje programske kode.

3.3.1 Netbeans IDE 8.0.2

Programsko okolje Netbeans IDE 8.0.2 [23] je bilo razvito v programskem jeziku Java. Uporaba programskega okolja je možna na vseh sistemih, ki podpirajo Javo (npr. Microsoft Windows, Mac OS X, Linux, Solaris, ...). Poleg pisanja programov v Javi podpira še pisanje programske kode v številnih drugih programskih jezikih (npr. PHP in C/C++).

3.3.2 Notepad++

Nodepad++ [22] je zastonjski urejevalnik izvorne kode. Program je podprt v različnih jezikih. Napisan je bil v programskem jeziku C++ in izdan pod licenco GPL. Velja za nepogrešljiv pripomoček vsakega programerja.

Program omogoča nameščanje številnih vtičnikov, ki popestrijo uporabniško izkušnjo. Programerju omogoča pregled celotnega besedila, iskanje določenih delov v besedilu, zamenjavo določenih nizov, primerjavo dveh besedil, črkovalnik,

skrivanje ali odkrivanje določenih delov besedila in številne druge funkcionalnosti. Dve zelo koristni funkciji programa sta zamikanje in obarvanje sintakse določenega programskega jezika.

3.3.3 SQL Server 2014 Management Studio

SQL Server 2014 Management Studio [27] je programsko orodje, ki se uporablja za upravljanje, konfiguracijo in administracijo vseh komponent, ki se nahajajo znotraj MSSQL. Program vsebuje skriptni urejevalnik in grafična orodja, ki delajo z objekti in razširitvami na strežniku. Poleg tega omogoča razvoj, nadzor in nadgradnjo podatkovnih komponent, ki so uporabljene s strani aplikacije. To so lahko podatkovne baze ali podatkovna skladišča. Programsko okolje omogoča T-SQL, DMX in XML jezikovne urejevalnike za urejanje in razhroščevanje skript.

3.3.4 Adobe Illustrator

Adobe Illustrator [2] je profesionalni grafični program za risanje vektorskih grafik. Razvilo ga je podjetje Adobe Systems. Prva verzija programa je izšla leta 1987. Program je bil narejen v programskem jeziku C++. Odtod izhaja njegova zmožnost hitrega izvajanja programa. Trenutno je podprt s strani dveh operacijskih sistemov Mac OS X in Microsoft Windows. Izjemno dobre rezultate daje pri ustvarjanju logotipov.

Bogat grafični vmesnik ponuja za delo številna orodja, ki olajšajo delo vsakemu oblikovalcu. Če želimo obstoječe funkcionalnosti nadgraditi lahko prenesemo vtičnike, ki nam to omogočajo.

Eden izmed zelo pomembnih orodji je pero. S pomočjo tega orodja lahko oblikujemo predmet. To dosežemo tako, da povežemo posamezne točke. Dobljene krivulje lahko različno upogibamo, brišemo ali pa dodajamo točke na krivuljo.

Nekaj osnovnih orodij, ki so še zelo koristna za delo, in sicer: radirka (uporablja se za brisanja posameznih delov na platnu), besedilo (za pisanje

besedila), pravokotnik (za oblikovanje pravokotnikov različnih velikosti).

Po končanem ustvarjanju v programu lahko shranimo delo v primarni format `Ai` ali pa v kakšen drugi.

3.3.5 Adobe Photoshop

Adobe Photoshop [1] je program za obdelavo fotografij in drugih grafik. S pomočjo programa lahko oblikujemo sliko ali druge objekte na platnu. Sliki oziroma objektu na platnu lahko dodajamo različne plasti, maske, spreminjamo barvne modele in še številne druge lastnosti.

Ob zagonu programa se nam prikaže menijska vrstica, v kateri se nahajajo osnovna orodja za delo (radirka, čarobna paličica, orodje za ostrenje, besedilo, itd.). Razširitve posameznih funkcij so možne z namestitvijo vtičnikov. Program uporablja za shranjevanje primarni format `psd`, nastalo delo pa je možno shraniti tudi v druge formate.

Poglavje 4

Razvoj aplikacije

V četrtem poglavju bomo opisali sestavo, funkcionalnost podatkovne baze in dvonivojsko arhitekturo, na kateri je zasnov naš prototip sistema. Podrobno bomo opisali programske in oblikovne rešitve pri izdelavi namizne aplikacije. Na koncu se bomo dotaknili tudi analize in testiranja aplikacije.

4.1 Implementacija podatkovne baze

V tem podpoglavju se bomo osredotočili na postopek ustvarjanja podatkovne baze, poznamo dva načina. Opisali bomo entite, ki sestavljajo našo podatkovno bazo, shranjevalno preceduro in postopek vzpostavljanja povezave med aplikacijo in podatkovno bazo.

4.1.1 Ustvarjanje baze

Podatkovna baza je zbirka med seboj pomensko povezanih podatkov, ki so shranjeni v računalniškem sistemu. Dostop do njih je centraliziran in omogočen s pomočjo sistema za upravljanje podatkovnih baz.

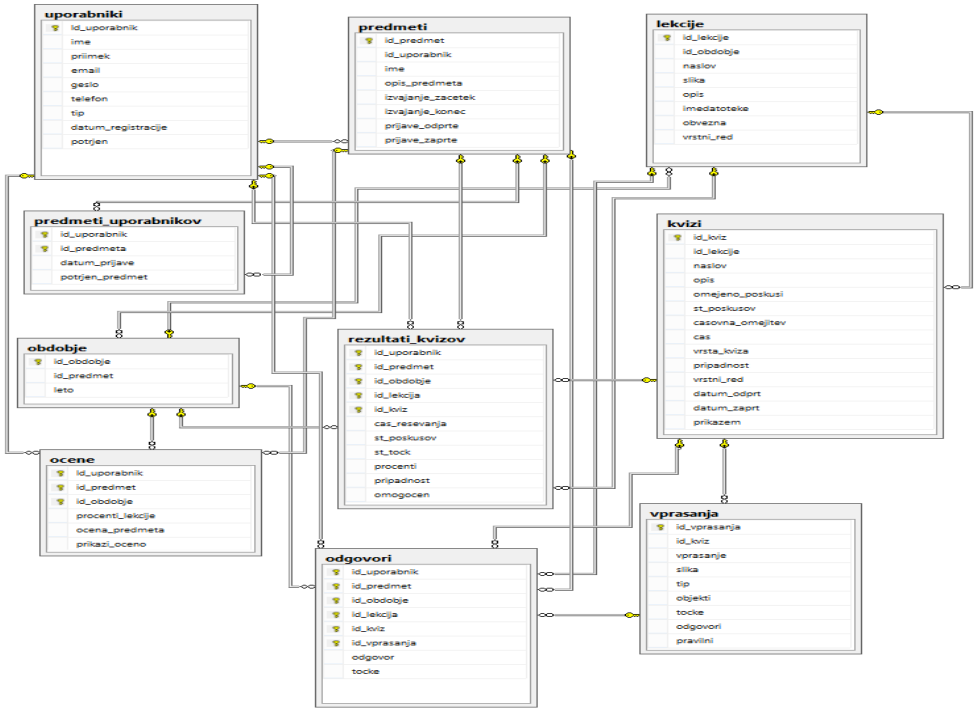
Podatkovno bazo smo ustvarili v programskem okolju SQL Server 2014 Management Studio. Novo podatkovno bazo ustvarimo tako, da pritisnemo na mapo `Databases` z desnim pritiskom gumba na miški in izberemo `New Database`. Podatkovno zbirko smo poimenovali `ogrodje`. Ustvarili jo bi

lahko tudi tako, da bi izvedli spodaj napisane ukaze:

```
1 use master
2 go
3 create database ogrodje
```

4.1.2 Entite

Entitete so objekti (osebki) iz realnega sveta o katerih zbiramo, obdelujemo in hranimo podatke ter informacije. Prikaz posameznih entite in povezav med njimi prikazuje diagram na sliki 4.1.



Slika 4.1: Prikaz diagrama podatkovne baze.

Naša podatkovna baza je sestavljena iz 10 entit. Entite, ki jih vsebuje naša podatkovna baza so:

- Entiteta `uporabniki` - hrani podatke o uporabnikih. Podakti, ki jih hranimo so: ključ uporabnika, ime, priimek, elektronska pošta, telefon,

vrsta uporabnika (atribut tip), datum registracije in potrjenost. Vse uporabnike lahko vidi skrbnik aplikacije, uporabnike svojih predmetov pa učitelj.

- Entiteta `predmeti` - hrani podatke o predmetih, ki jih učijo učitelji. Poleg podatkov o imenu in opisu predmeta se beležijo še podatki o obdobju izvajanja predmeta ter obdobju vpisa v predmet.

- Entiteta `predmeti_uporabnikov` - hrani predmete učencev. Poleg dveh primarnih ključev (`Id_uporabnik` in `id_predmet`) se beležita še podatka o datumu zahteve za vpis in potrjenost predmeta.

- Entiteta `obdobje` - hrani podatke o obdobjih za vsak predmet. V primeru, da želi učitelj poučevati predmete v novem letu lahko to stori tako, da ustvari novo obdobje.

- Entiteta `lekcije` - v tej tabeli hranimo podatke o lekcijah. Lekcije za svoj predmet lahko dodaja samo učitelj. Entiteta je povezana s tabelo `obdobje`.

- Entiteta `kvizi` - hrani podatke o kvizih. Vsaka lekcija lahko ima enega ali več kvizov. Posamezne kvize lahko dodaja samo učitelj istega predmeta.

- Entiteta `vprasanja` - v tabeli so shranjena vprašanja za vsak kviz. Kviz lahko ima eno ali več vprašanj. Vprašanja lahko dodaja samo učitelj.

- Entiteta `odgovori` - v tej tabeli so shranjeni vsi odgovori učencev. Tabela se uporablja pri točkovanju in preverjanju pravilnih odgovorov. Dostop do posameznih odgovorov v tabeli ima samo učitelj.

- Entiteta `rezultati_kvizov` - hrani podatke o vseh kvizih, ki so jih reševali učenci. Tabela je vidna samo učiteljem.

- Entiteta `ocene` - v njej so shranjeni podatki o vseh ocenah učencev. Vsak učitelj ima pregled nad ocenami svojih učencev.

Entite so med seboj povezane s pomočjo identifikacijskih ključev. Primarni ključ enolično določa vrstico v tabeli. To pomeni, da se vrednosti katerega-koli primarnega ključa med seboj razlikujeta. Primer ustvarjanja tabele, ki vsebuje primarni ključ:

```
create table uporabniki
(
    Id_uporabnik bigint identity primary key,
    ime nvarchar(MAX) not null,
    priimek nvarchar(MAX) not null,
    email nvarchar(MAX) not null,
    geslo nvarchar(MAX) not null,
    telefon nvarchar(MAX) null,
    tip int not null,
    datum_registracije datetime not null,
    potrjen bit not null
);
```

Če pride do prekrivanja vrednosti, sistem sam javi napako. Posamezno vrednost primarnega ključa lahko poenostavimo s spodnjim ukazom:

```
dbcc checkident('uporabniki', RESEED, 0);
```

Ukaz poenostavi vrednost primarnega ključa na nič. Ob ponovnem dodajanju podatkov v tabelo bo vrednost primarnega ključa 1.

Tuji ključ v tabeli je identifikator, ki kaže na primarni ključ, s katerim je tabela v razmerju. Primer ustvarjanja tabele `kvizi`, ki vsebuje poleg primarnega ključa še tuji ključ:

```
create table kvizi(
   id_kviz bigint identity primary key,
   id_lekcije bigint not null,
   naslov nvarchar(MAX) not null,
   opis nvarchar(MAX) not null,
   omejeno_poskusi bit not null,
   st_poskusov int null,
   casovna_omejitev bit not null,
   cas time(0) null,
   vrsta_kviza int not null,
   pripadnost nvarchar(MAX) not null,
   vrstni_red int null,
```

```
13    datum_odprt date not null,
14    datum_zaprt date not null,
15    prikazem bit not null,
16    constraint foreign key (id_lekcije)
17    references lekcije(id_lekcije)
18    on delete cascade
19    on update cascade
20 );
```

Tujemu ključu lahko dodamo omejitev. Ukaza `on delete cascade` in `on update cascade` pomenita, da lahko izbrišemo oziroma spremenimo vrednost vrstice, če pride do sprememb v tabeli, ki vsebuje ta primarni ključ.

4.1.3 Shranjena procedura

Shranjena procedura je podprogram, namenjen aplikacijam, ki dostopajo do relacijskega podatkovnega sistema. Shranjena je v podatkovni bazi. Uporaba shranjene procedure je smiselna kadar imamo opravka s pogojnimi stavki, izvajanji več različnih stavkov SQL, vračanji vrednosti itd. Shranjena procedura vrne množico rezultatov. Aplikacija kliče shranjeno proceduro z ukazom `Call`, ta funkcija lahko vsebuje tudi parametre. V MSSQL se izvede s pomočjo ukaza `execute`.

```
1  create procedure procenti_lekcije_ucenec
2  @id_le bigint,
3  @id_up bigint
4  AS
5  set NOCOUNT ON;
6  declare @st_kv int
7    set @st_kv=(select count(k.id_kviz)
8    from kvizi as k
9    where k.id_lekcije=@id_le and k.pripadnost='-1')
10 declare @res_le float=(select sum(k.procenti)
11  from rezultati_kvizov as k
12  where k.id_lekcija=@id_le and k.pripadnost='-1'
13  and k.id_uporabnik=@id_up)
14 declare @v float=(@res_le/(@st_kv*100))*100
15 select @v as 'uspesnost lekcije v [%]'
```

Procedura `procenti_lekcije_ucenec` sešteje vse kvize določene lekcije, ki jih je učenec rešil. Rezultat, ki ga vrne, je decimalno število. Ukaz `declare`

nam omogoči, da definiramo svojo spremenljivko. S pomočjo ukaza `set nocount on;` [25] izključimo obvestila o spremenjenih/dodanih vrsticah in tako zelo pohitrimo delovanje procedure.

4.1.4 Povezava s podatkovno bazo

JDBC je gonilnik, ki nam omogoča vzpostavitev povezave med javansko aplikacijo in podatkovno bazo [17]. Gonilnik podpira MSSQL, SQL Azure in paralelna podatkovna skladišča. Primer vzpostavljanja povezave z našo podatkovno bazo `ogrodje`:

```
1 try {
2  Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
3  con=DriverManager.getConnection("jdbc:sqlserver://localhost:1433;"+
4  "databaseName=ogrodje;integratedSecurity=true;");
5 } catch (ClassNotFoundException | SQLException ex) {
6       System.out.println(ex.getMessage());
7 }
```

S pomočjo ukaza `Class.forName` kličemo gonilnik. Razred vrste `DriverManager` dostopa do funkcije `getConnection`, ki služi za vzpostavitev povezave. Parameter v tej funkciji je povezovalni niz. S pomočjo vrednosti tega niza lahko vzpostavimo povezavo. Če je med vzpostavitvijo povezave prišlo do napake, jo izpišemo. Eden izmed parametrov, ki ga nastavimo v povezovalnem nizu, je tudi `integratedSecurity`. Nastavili smo ga na vrednost `true`, kar pomeni, da bo gonilnik uporabil sistemske povernilnice za identifikacijo. V nasprotnem primeru je potrebno vnesti uporabniško ime in geslo.

4.2 Dvonivojska arhitektura

Aplikacija deluje po principu dvonivojske arhitekture (slika 4.2). Za dvonivojsko arhitekturo je značilno, da imamo na eni strani enega ali več odjemalcev, na drugi strani pa podatkovni strežnik. Odjemalec in strežnik sodelujeta med seboj s pomočjo komunikacijskega omrežja. Odjemalec s pomočjo komunikacijskega omrežja pošlje zahtevo po podatkih, strežnik pa mu pošlje

zahtevane podatke nazaj. Program, ki omogoča in nadzoruje dostop do podatkovne baze je nameščen na podatkovnem strežniku (v našem primeru na MSSQL). Odjemalec za pošiljanje svojih zahtev uporablja aplikacijo. Ta vsebuje že vnaprej napisane ukaze za pošiljanje zahtev.

Odjemalec uporablja uporabniški vmesnik, pošilja zahteve po podatkih in izvaja njihovo procesiranje.

Podatkovni strežnik shranjuje podatke v podatkovno bazo, izvaja shranjevalne procedure, omogoča integriteto, varnost in zaklepanje podatkov. Podatek zaklene v primeru sočasnega dostopa.

Slika 4.2: Prikaz dvonivojske arhitekture.

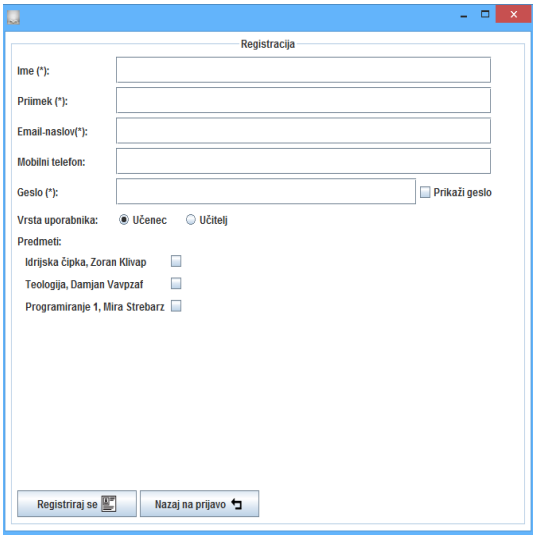
4.3 Registracija, prijava in pozabljeno geslo

Naslednja podpodpoglavja podrobno opisujejo postopek registracije, prijave in pošiljanje pozabljenega gesla.

4.3.1 Registracija

Registracija je namenjena učiteljem in učencem. Pri registraciji je potrebno izpolniti vsa obvezna vnosna polja. Obrazec vsebuje tudi nekaj neobveznih vnosnih polj, ki jih uporabniku ni potrebno izpolniti. Vsi napisi označeni z zvezdico predstavljajo obvezne podatke, ki jih mora učenec ali učitelj vnesiti za uspešno registracijo (slika 4.3).

Učenec lahko izbere predmete, ki jih želi obiskovati. Če je zamudil rok za vpis v predmet, lahko zahtevo za vpis pošlje, ko se prijavi v aplikacijo.



Slika 4.3: Zaslonska maska registracije za učenca.

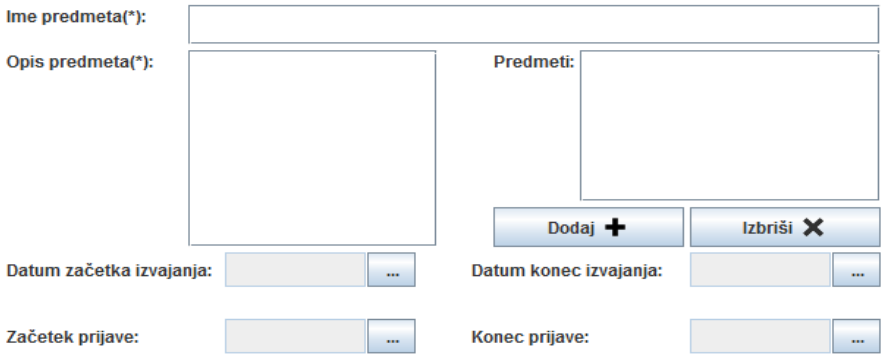
Gesla zakodiramo s pomočjo algoritma MD5 in soli. Algoritem MD5 je kodirna funkcija s 128-bitnim izhodom. Z vidika varnosti ne velja za najbolj varen algoritem, saj se da ugotoviti vrednosti nekaterih nizov s pomočjo mavričnih tabel. Mavrična tabela je tabela, ki vsebuje vnaprej izračunane zgoščevalne vrednosti in gesla. Nepridipravom, ki uporablajo mavrične tabele in druge metode za ugotavljanje gesel, utežimo delo tako, da dodamo geslu poljuben niz. Ta niz zakodiramo skupaj z geslom. Na ta način dobimo novo geslo, ki ga shranimo v podatkovno bazo. Uporabniku se tako doda poleg njegovega gesla še niz, ki se imenuje sol. Skupna vrednost (vrednost gesla in soli) se nato zakodira in primerja z vrednostjo v podatkovni bazi. Prijava je uspešna, če se vrednosti ujemata. Primer funkcije, ki zakodira niz s pomočjo algoritma MD5 in soli:

```
1 private String kodiranje(String geslo) {
2
3   String md5 = null;
4   if (geslo == null) {
5                 return null;
6   }
```

```
7    geslo = geslo + sol;
8    try {
9
10         MessageDigest digest = MessageDigest.getInstance("MD5");
11         digest.update(geslo.getBytes(), 0, geslo.length());
12         md5 = new BigInteger(1, digest.digest()).toString(16);
13
14   } catch (NoSuchAlgorithmException e) {
15          System.out.println(e.getMessage());
16   }
17   return md5;
18 }
```

Funkcija `kodiranje` pričakuje ob njenem klicu en parameter vrste `String`. Na začetku je vrednost niza `md5` nastavljena na `null`. Vrednost `null` pomeni neznano vrednost. V četrti vrstici se izvede pogojni stavek, če je vrednost spremenljivke `geslo` enako `null`. V primeru izvedbe pogojnega stavka vrnemo vrednost `null`. Sedma vrstica doda nizu `geslo` poleg svoje vrednosti, še vrednost soli. V bloku `try/catch` ustvarimo objekt vrste `MessageDigest`. Enajsta vrstica posodobi niz v objekt vrste `MessageDigest`. V naslednji vrstici pretvorimo dobljeni niz v šestnajstiški zapis. V primeru napake jo izpišemo in vrnemo niz.

Učitelj poleg zahtevanih podatkov doda še predmete, ki jih bo poučeval (slika 4.4). Pri dodajanju predmeta navede ime, opis, obdobje izvajanja in obdobje vpisa v predmet.



Slika 4.4: Dodatna polja za učitelja pri registraciji.

4.3.2 Prijava

Uporabnik se prijavi s pomočjo elektronske pošte in gesla (slika 4.5). Vrsta uporabnika določa, kam bo uporabnik preusmerjen po uspešni prijavi. V primeru napačnega ali manjkajočega vnosa podatkov v vnosna polja, se uporabnika obvesti. Uporabnik lahko izvede prijavo tako, da pritisne na gumb `Prijava` ali tipko `Enter`.

Prijava v aplikacijo lahko traja dlje časa, zato je potrebno uporabiti nit. V programskem jeziku Java nam to omogoča kar sam `swing`. Swing ponuja implementcijo množice komponent, ki so namenjene za delo z grafičnimi vmesniki. Ponuja pa tudi abstraktni razred za delo z nitmi, ki se imenuje `SwingWorker` [26]. Pri razširitvi razreda je potrebno uporabiti vsaj dve metode, in sicer `doInBackground` in `done`. Metoda `doInBackground` opravlja delo, ki ga je potrebno izvesti v ozadju. Metoda `Done` pa se izvede, ko metoda `doInBackground` konča svoje delo. Nit izvedemo s pomočjo metode `execute`.

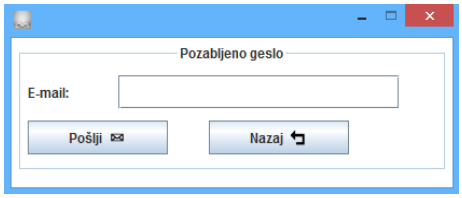
Uporabnik se lahko tudi odjavi, ko to želi. Pri odjavi se sprostijo vsi viri, ki jih drži aplikacija. Vsa okna, ki so odprta se zaprejo in vrednosti vseh pomembnih spremenljivk se nastavijo na začetne vrednosti.

Slika 4.5: Prijava uporabnika.

4.3.3 Pozabljeno geslo

V primeru, da je uporabnik pozabil geslo in že opravil registracijo, lahko s pomočjo obstoječe elektronske pošte pošlje zahtevo za geslo (slika 4.6). Geslo, ki se pošlje uporabniku je privzeto in ga lahko spremeni. V primeru

napačnega vnosa elektronske pošte se uporabnika obvesti s pomočjo dialoga.



Slika 4.6: Prikaz zaslona za pošiljanje pozabljenega gesla.

4.4 Nadzorna plošča skrbnika

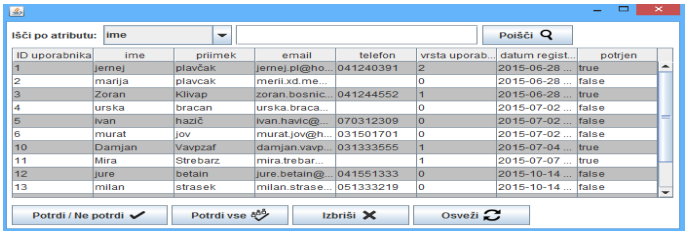
V tem delu bomo spoznali glavne funkcionalnosti skrbnika. Vsak skrbnik lahko ureja svoje osebne podatke, potrjuje uporabnike, dodaja ali ureja predmete učiteljev. Grafični vmesnik skrbnika je prikazan na sliki 4.7.

Slika 4.7: Prikaz nadzorne plošče skrbnika.

4.4.1 Potrjevanje novih uporabnikov

Novi uporabniki so na začetku nepotrjeni. Glavna vloga skrbnika je, da potrdi uporabnike (slika 4.8). Uporabnike je možno poiskati po določenem atributu. Atributi, ki olajšajo iskanje uporabnikov ali določenega uporabnika

so: ime, priimek, elektronska pošta, telefon, datum registracije in potrjenost. Rezultat iskanja so uporabniki, ki ustrezajo kriteriju.



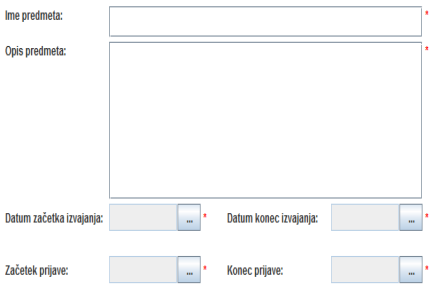
Slika 4.8: Prikaz zaslonske maske potrjevanja uporabnikov.

4.4.2 Dodajanje ali urejanje predmetov

V primeru, da si želi učitelj učiti dodaten predmet, to javi skrbniku (slika 4.9). Skrbnik lahko tako doda željeni predmet učitelju. Pri dodajanju mora skrbnik pravilno vnesiti podatke v vnosna polja. V primeru nepravilnosti se zraven vnosnega polja ali katerega drugega gradnika prikaže rdeče obarvana zvezdica. Pritisk na to zvezdico nam v orodju za namig izpiše vzrok napake. V nadaljevanju pa si bomo pogledali del kode, ki izpiše napako v orodju za namig:

```
1  boolean izbranucitelj = false;
2  for (int i = 0; i < uciteljidadminrd.size(); i++) {
3
4    if (uciteljidadminrd.get(i).isSelected()) {
5          izbranucitelj = true;
6          break;
7    }
8
9  }
10 if (!izbranucitelj) {
11
12   napizberiucitelja.setText("*");
13   napizberiucitelja.setForeground(rdeca);
14   napizberiucitelja.setCursor(new Cursor(Cursor.HAND_CURSOR));
15   napizberiucitelja.setToolTipText("Izberi ucitelja");
16
17 }
```

Vrednost spremenljivke `izbranucitelj` je na začetku napačna. V naslednjem koraku se s pomočjo `for` zanke sprehodimo skozi seznam učiteljev. Pogojni stavek se izpolni, če je katerikoli učitelj izbran. Izpolnjen pogoj nastavi spremenljivko `izbranucitelj` na resnično in povzroči izhod iz zanke. Deseta vrstica se začne s pogojnim stavkom in se izvede v primeru, če je vrednost spremenljivke `izbranucitelj` napačna. Izpolnjen pogoj v deseti vrstici nastavi besedilo spremenljivke `napizberiucitelja` na zvezdico, barvo pisave spremeni na rdečo in miškin kazalec v ročico.



Slika 4.9: Prikaz napak pri dodajanju predmeta. Rdeča zvezdica ponazarja manjkajoče podatke oz. napačne vnose v vnosna polja.

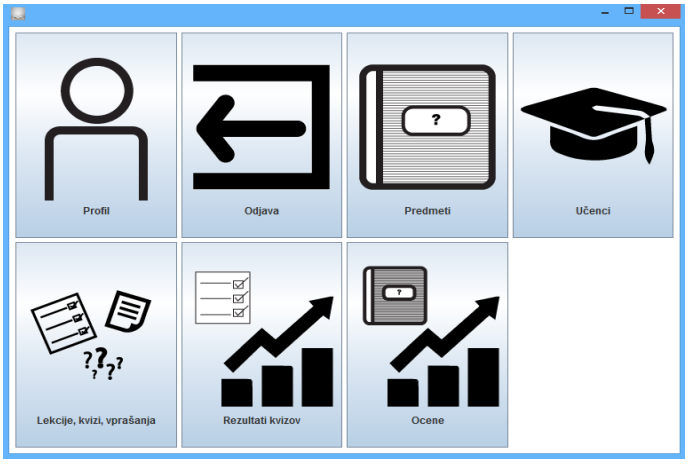
4.5 Nadzorna plošča učitelja

V tem podpoglavju bomo podrobno opisali funkcionalnosti učitelja. Učitelj lahko opravlja vse funkcionalnosti šele, ko ga potrdi skrbnik aplikacije. Če učitelj ni potrjen, lahko ureja samo svoje osebne podatke. Grafični vmesnik, ki predstavlja nadzorno ploščo učitelja, vsebuje šest gumbov, ki sprožijo prikaz posameznega okna. Primer programske kode ustvarjenja gumba in dogodka, ki se zgodi ob pritisku na gumb:

```
1 JButton gumbuciteljprofil = new JButton(uporabnikiicon);
2  gumbuciteljprofil.setText("Profil");
3  gumbuciteljprofil.setVerticalTextPosition(SwingConstants.BOTTOM);
4  gumbuciteljprofil.setHorizontalTextPosition(SwingConstants.CENTER);
5  gumbuciteljprofil.setFocusPainted(false);
6  gumbuciteljprofil.setCursor(new Cursor(Cursor.HAND_CURSOR));
```

```
7  gumbuciteljprofil.addActionListener(new ActionListener() {
8          @Override
9          public void actionPerformed(ActionEvent e) {
10                 if (zastavicaprofilucitelja) {
11                         oknoUciteljProfil();
12                 }
13         }
14 });
```

Gumbu (slika 4.10) lahko dodamo besedilo, postavitev besedila, ikono, obarvano žarišče okoli gumba, vrsto kazalca miške in še številne druge lastnosti. Poslušalca dogodkov dodamo s pomočjo funkcije `addActionListener`. Znotraj funkcije je definiran razred `ActionListener`. Ta razred implementira funkcijo `actionPerformed`, ki zazna vsak pritisk na gumb. Ob pritisku na gumb se preveri vrednost pogojnega stavka, če je vrednost resnična se izvede funkcija znotraj njega.



Slika 4.10: Prikaz nadzorne plošče učitelja

4.5.1 Urejanje predmetov in potrjevanje učencev

Učitelj lahko uči enega ali več predmetov. Predmete lahko ureja, ne more pa jih dodajati.

Vsak učitelj ima pregled nad številom svojih učencev. Učence, ki obiskujejo njegove predmete lahko išče po naslednjih atributih: `id` predmeta, ime

predmeta, `id` uporabnika, ime, priimek, elektronska pošta, telefon, potrjenost in datum vpisa v predmet. Vsakega učenca lahko vpiše v predmet ali izpiše.

4.5.2 Dodajanje lekcij, kvizov in vprašanj

V naslednjih razdelkih bomo opisali dodajanje lekcij, kvizov in vprašanj. Pri opisovanju se bomo osredotočili tudi na posamezna vnosna polja.

Dodajanje lekcij

Pri dodajanju lekcij mora učitelj pravilno izpolniti obvezna vnosna polja: naslov, opis in vrstni red (kako si lekcije sledijo). Dodati ali izbrati je potrebno obdobje izvajanja predmeta. Lekcija je lahko obvezna in vključuje tudi sliko. Učna snov lekcije je `pdf` datoteka, ki predstavlja celotno vsebino lekcije.

Ko je učitelj pravilno vnesel vse podatke, lahko ustvari lekcijo, kasneje pa jo lahko tudi uredi (v primeru, da ni zadovoljen z vnešenimi podatki).

Dodajanje kvizov

Obvezna polja pri dodajanju kviza so: naslov, opis, vrsta (začetni, osnovni, podkviz in test), datum izvajanja in naslov lekcije. Začetni kviz predstavlja prvi kviz v lekciji. Ponavadi mu sledijo osnovni kvizi. Osnovni kviz lahko vsebuje nič ali več podkvizov. V primeru, da učenec ne doseže polovico števila točk, se ga preusmeri na podkvize. Namen podkvizov je, da učenec utrdi svoje znanje, preden napreduje nazaj na neuspešno rešen osnovni kviz.

Neobvezna polja so: časovna omejitev, število poskusov ter prikaz kviza. Čas trajanja kviza učitelj določi tako, da nastavi ure, minute in sekunde. Največje število poskusov je 100, najmanjše pa 1. Potrditveno polje `Prikaz kviza` omogoči vidnost kviza.

Kviz ustvari tako, da pritisne na gumb `Shrani`. Prikaže se mu dialog s statusom, ki pove ali je kviz uspešno ustvarjen ali ne.

Dodajanje vprašanj

Poleg samega vprašanja lahko učitelj doda tudi sliko. Odloči se lahko za slikovni ali besedilni prikaz odgovorov. Poleg tega lahko izbira med gradniki, ki bodo prikazali dani odgovor. Ti gradniki so: besedilno polje (ne moremo ga uporabiti, če izberemo sliko), izbirna tipka in potrditveno polje. Za vsako vprašanje mora učitelj določiti tudi število točk. Na koncu mora obvezno izbrati obstoječi kviz, kateremu bo pripadalo vprašanje.

Vprašanje doda h kvizu, s pritiskom na gumb `Shrani`. Če so prisotne napake jih je potrebno odpraviti. Uspešnost akcije shranjevanja je prikazana v dialogu.

4.5.3 Rezultati kvizov

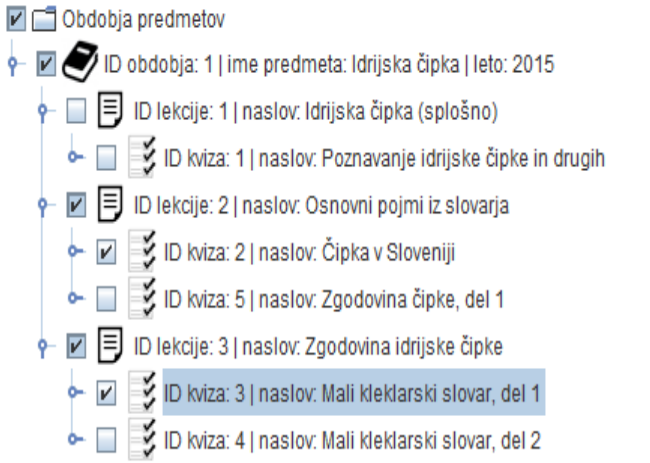
Učitelj ima pregled nad rezultati vseh kvizov, ki so jih reševali učenci. Za vsak kviz lahko vidi odgovore zadnjega reševanja. Poleg tega pa še število poskusov, čas reševanja, doseženo število točk in uspešnost v odstotkih najuspešnejšega kviza. Aplikacija omogoča učitelju uporabo grafa. Izbira lahko med stolpičnim in tortnim grafom. Graf prikazuje uspešnost učencev v nekem obdobju pri določenih lekcijah ali kvizih. Gumb z napisom `Uspešnost posameznika` omogoča učitelju prikazati uspešnost vsakega učenca (seštevek vseh obveznih lekcij skupaj v določenem obdobju).

4.5.4 Kopiranje obdobij

Učitelj lahko izbere posamezne lekcije, kvize, vprašanja iz različnih obdobij ali celega obdobja, in jih prekopira v novo obdobje. Na ta način učitelju olajšamo delo sestavljanja lekcij, kvizov in vprašanj za naslednje leto. Za prikaz obdobij, lekcij, kvizov in vprašanj smo uporabili drevesno strukturo. Drevesna struktura vsebuje vozlišča, ki vsebujejo podatke. Ti podatki so razporejeni po navpični osi navzdol. Na vrhu drevesne strukture se nahaja začetno vozlišče. Vozlišča, ki imajo potomce in niso začetna vozlišča, imenujemo veje. Vozlišče, ki nima potomcev in ni začetno vozlišče pa list. Vsaka

drevesna struktura ima začetno vozlišče, ki ga lahko prikažemo ali ne.

Na sliki 4.11 vidimo začetno vozlišče, ki vsebuje napis `Obdobja predmetov`. Za prikaz besedila smo izbrali potrditvena polja, ki vsebujejo tudi ikone. Potrditvena polja lahko obkljukamo ali pa tudi ne. Razred v Javi, ki uporablja drevesno strukturo se imenuje `JTree` [14]. Vozlišča, ki jih izbere uporabnik dobimo z uporabo metode `getSelectionPaths`.



Slika 4.11: Prikaz uporabe drevesne strukture.

4.5.5 Ocenjevanje

Učitelj pridobi ocene učencev s pomočjo tabele `rezultati_kvizov`. Vsako pridobljeno oceno učencev lahko uredi in nastavi, da bo vidna učencu. Pri ocenjevanju se neobvezne lekcije ne upoštevajo. Odstotke končne ocene dobimo po naslednji enačbi:

$$\text{uspešnost učenca}(\%) = \frac{\text{seštevek vseh rešenih lekcij (obveznih)}}{\text{št. vseh obveznih lekcij } 100} 100$$

Na podlagi te enačbe dobimo tudi končno oceno. Meje za posamezne ocene so prikazane na sliki 4.12.

Učitelj lahko izpiše ocene posameznih učencev v formatu datoteke excel. Preden izvede korak izpisa, ga program vpraša kam želi shraniti excel da-

ocena	%
$1 \leq 5$	$0 < 50$
6	$51 \leq 60$
7	$61 \leq 70$
8	$71 \leq 80$
9	$81 \leq 90$
10	$91 \leq 100$

Slika 4.12: Ocenjevalna tabela.

toteko in kako jo bo poimenoval. Izpis excel datoteke vsebuje poleg ocen učencev še povprečno oceno predmeta ter datum izpisa.

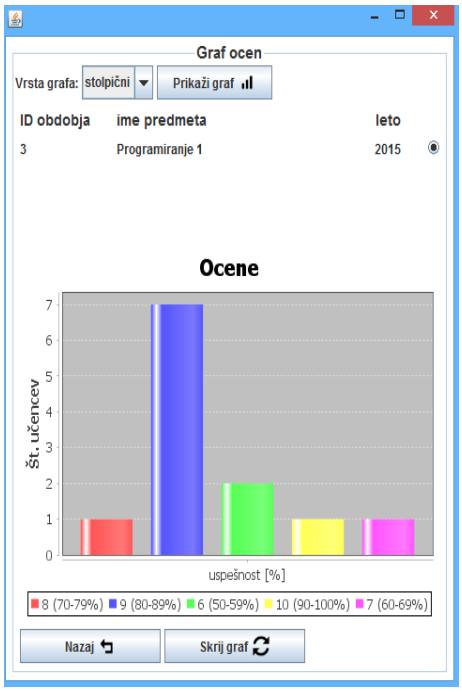
4.5.6 Grafi

Rezultate kvizov in ocene posameznih učencev je možno prikazati s pomočjo grafov (slika 4.13). Učitelj lahko izbira med tortnim in stolpičnim grafom. Knjižnjica, ki nam omogoča delo z grafi v Javi se imenuje `JFreeChart` [13]. Različica 1.0.19 je odprtokodna in zastonjska. Omogoča prilagodljiv videz, kar pomeni, da je primeren za različne dimenzije zaslonov. Podpira različne izhodne tipe med katerimi so tudi Swing in JavaFX komponente, slikovni formati (`jpg` in `png`) in formati, ki podpirajo vektorsko grafiko (`pdf`, `eps` in `svg`). Izhodni tipi nam omogočajo shranjevanje grafov za nadaljno uporabo. Primer programske kode, ki ustvari tortni graf:

```
1  private JPanel pitinGrafPanel(HashMap<String, Integer> pod,
2                  String naslov) {
3
4   DefaultPieDataset dataset = new DefaultPieDataset();
5    for (Map.Entry<String, Integer> entry : pod.entrySet()) {
6          dataset.setValue(entry.getKey(), entry.getValue());
7   }
8
9   JFreeChart chart = ChartFactory.createPieChart(
10                 naslov, // naslov grafa
11                 dataset, // podatki
12                 true, // vkljucim legendo
13                 true,
14                 false);
```

```
15    return new ChartPanel(chart);
16
17 }
```

Funkcija `pitinGrafPanel` vsebuje dva parametra: slovar in niz. Slovar vsebuje ključ, ki je tipa `String` (niz). Poleg ključa pa vsebuje še vrednost, ki je tipa `Integer` (celo število). Slovar uporabimo takrat, ko imamo podatke urejene po geslih ali ključih. S pomočjo slovarja smo določili nabor podatkov za tortni graf. Prva vrednost funkcije `setValue` je ključ slovarja, druga pa število podatkov. Graf smo ustvarili tako, da smo nastavili ustrezne vrednosti v funkciji `createPieChart`. Funkcija zahteva naslednje parametre: naslov grafa, podatkovni nabor, vključevanje legende, orodja za namige (angl. tooltip) in lokalne nastavitve. Ko je graf ustvarjen, ga je potrebno prikazati v panelu za grafe, kar nam omogoča razred `ChartPanel`.



Slika 4.13: Prikaz ocen učencev pri predmetu Programiranje 1.

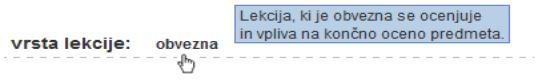
4.6 Grafični vmesnik za učence

Učencu se ob prijavi prikaže zaslon z njegovimi predmeti. Vsebina določenega predmeta je vidna učencu šele, ko je potrjen s strani učitelja. Zavihek `Profil` prikazuje osebne podatke učenca, ki jih lahko spreminja. Poleg spreminjanja osebnih podatkov ima možnost, da se vpiše v predmet, katerega rok za vpis je zamudil ob regitraciji računa. Vsebine predmeta zajemajo lekcije, ki jih mora učenec uspešno rešiti, da opravi predmet. Vsaka lekcija vsebuje `pdf` datoteko, ki zajema znanje lekcije. Učenca se glede na doseženo število točk preusmeri na naslednji kviz. Ko uporabnik uspešno reši vse kvize, lahko napreduje na naslednjo lekcijo. Seštevek vseh obveznih lekcij predstavlja končno oceno učenca. Učenec ima vpogled v svoje ocene v razdelku za ocene.

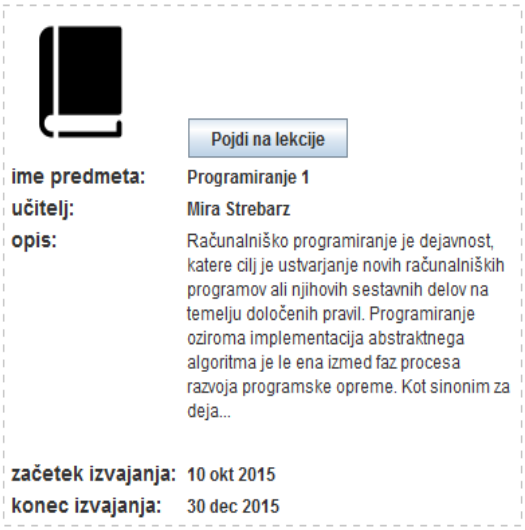
4.6.1 Vpogled v lekcijo

Zavihek `Predmeti` vsebuje vse predmete, ki jih je izbral učenec. Panel predmet (slika 4.15) vsebuje podatke o predmetu: ime predmeta, opis, ime učitelja in čas izvajanja. Ob pritisku na gumb `Pojdi na lekcije` se nam prikažejo lekcije ali samo začetna lekcija. Začetna lekcija se nam prikaže, če še nismo začeli reševati kvizov lekcij. Vsaka lekcija (slika 4.16) ima naslednje podatke: naslov, opis, prikazna slika (lahko jo doda učitelj ali pa se uporabi prevzeta), obveznost in uspešnost lekcije (podatki o uspešnosti se izpišejo ob prvem reševanju kvizov lekcije).

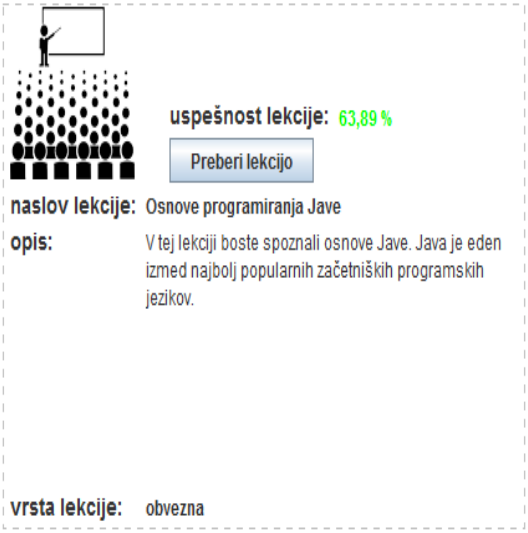
Namig (slika 4.14) se nam prikaže, če z miškinim kazalcem prekrijemo del besedila. Ko zagledamo miškin kazalec v obliki ročice vemo, da se nam bo prikazal namig. Besedilo v namigu lahko oblikujemo s pomočjo kaskadnih stilskih podlog. Prelom besedila v novo vrstico dosežemo z ukazom `<br>`.



Slika 4.14: Prikaz orodja za namige.



Slika 4.15: Primer prikaz panela predmet.



Slika 4.16: Primer prikaza panela lekcije.

Ob pritisku na gumb `Preberi lekcijo` se nam prikaže `pdf` datoteka. Prikaz `pdf` datoteke (slika 4.17) v Javi nam omogoča knjižnjica `icepdf`. To je odprtokodni mehanizem za pregled `pdf` dokumentov, ki ga lahko vgradimo tudi v javanske aplikacije [8]. Omogoča ustvarjanje in pretvorbo `pdf` datoteke v sliko (npr. `png`), pripravo dokumenta za tiskanje, iskanje po besedilu,

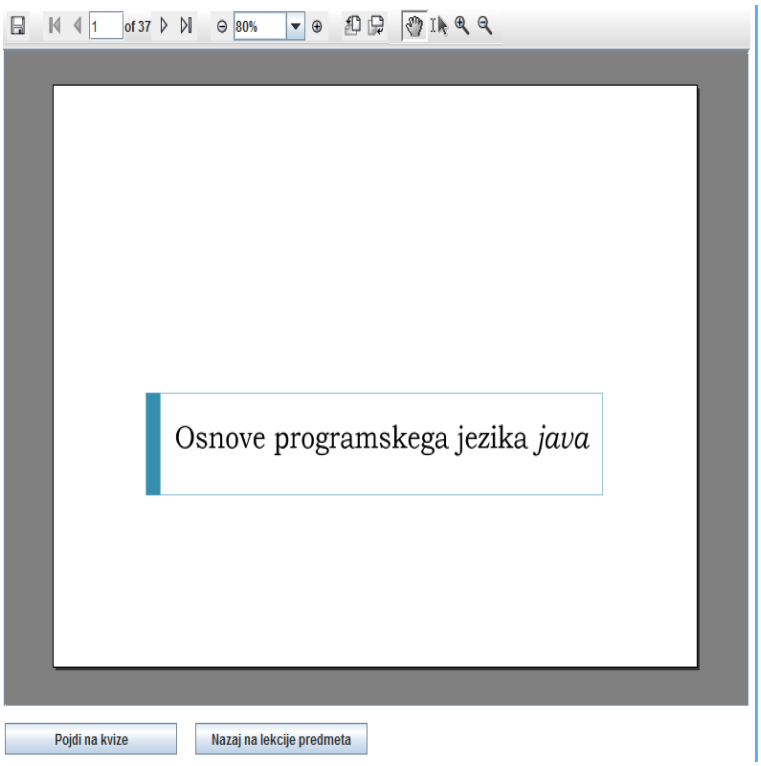
označevanje določenih besed ali stavkov v besedilu in interaktivne obrazce. Primer programske kode, ki prikaže `pdf` datoteko:

```
if (controller != null) {
                controller.dispose();
}
controller = new SwingController();

PropertiesManager p= new PropertiesManager(System.getProperties(),
ResourceBundle.getBundle(PropertiesManager.DEFAULT_MESSAGE_BUNDLE));

p.setBoolean(PropertiesManager.PROPERTY_SHOW_UTILITY_PRINT,
 Boolean.FALSE);
p.setBoolean(PropertiesManager.PROPERTY_SHOW_UTILITY_SAVE,
Boolean.TRUE);
p.setBoolean(PropertiesManager.PROPERTY_SHOW_TOOLBAR_ANNOTATION,
Boolean.FALSE);
p.setBoolean(PropertiesManager.PROPERTY_SHOW_UTILITY_UPANE,
 Boolean.FALSE);
p.setBoolean(PropertiesManager.PROPERTY_SHOW_UTILITY_SEARCH,
 Boolean.FALSE);

p.setBoolean(PropertiesManager.PROPERTY_SHOW_TOOLBAR_FIT,
 Boolean.FALSE);
p.setBoolean(PropertiesManager.PROPERTY_SHOW_STATUSBAR,
 Boolean.FALSE);

p.set(PropertiesManager.PROPERTY_DEFAULT_ZOOM_LEVEL, "0.8");
p.set(PropertiesManager.PROPERTY_ZOOM_RANGES, "0.5,0.6,
0.8,1,1.5,2");

factory = new SwingViewBuilder(controller, p);

viewerComponentPanel = factory.buildViewerPanel();

controller.getDocumentViewController().setAnnotationCallback(
new org.icepdf.ri.common.MyAnnotationCallback(
controller.getDocumentViewController()));
controller.openDocument(poiscidatoteke("lekcije",
           imedat, 1, id_obd, id_lek).getAbsolutePath());
```

Na začetku preverimo, če je spremenljivka `controller` enaka vrednosti `null`. V primeru, da je krmilnik že ustvarjen, sprostimo njegov vir. Razred `PropertiesManager` omogoča nastavljanje vrednosti, ki se uporabijo za prikaz `pdf` komponent. Omogoča nam nastavljanje določenih funkcionalnosti in

prikaz različnih komponent: ikona za tiskanje `pdf` datoteke, možnost shranjevanja, prikaz iskalnika, statusne vrstice, različna orodja za pomoč uporabnikom, nastavljanje zumiranja, itd. Razred `SwingViewBuilder` lahko sprejme do dva parametra. Prvi parameter je krmilnik, drugi pa vsebuje nastavitve, ki smo jih nastavili s pomočjo razreda `PropertiesManager`. Funkcija `buildViewerPanel` ustvari panel, ki bo prikazoval `pdf` datoteko in orodno vrstico.

Funkcija `setAnnotationCallback` omogoča razvijalcu nadzirati zaznambo in izvajanje njenih akcij. Razvijalec ima možnost spreminjati vrednosti atributov zaznambe. Preden se zaznamba izriše lahko spreminja obrobni slog, barvo in dolžino črte. Datoteko `pdf` odpremo s pomočjo funkcije `openDocument`. Funkcija prejme niz, ki predstavlja lokacijo `pdf` datoteke.



Slika 4.17: Prikaz vsebine lekcije.

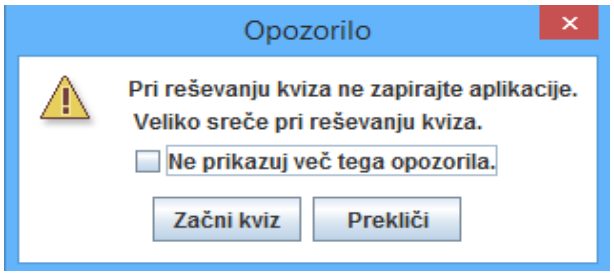
4.6.2 Kvizi

Lekcija vsebuje učno snov, ki je predstavljena v obliki `pdf` datoteke. Ko se učenec nauči snov lekcije, lahko svoje znanje preveri s pomočjo kvizov, ki jih vsebuje lekcija. Vsak kviz vsebuje naslednje podatke (na sliki 4.18): ime, opis, vrsto kviza, časovno omejitev, število poskusov, datum in čas izvajanja, najboljši čas, uspešnost kviza ter številu poskusov, ki so še na voljo za reševanje kviza.

Slika 4.18: Prikaz panela kviz.

Prekritje določenega dela besedila z miškinim kazalcem prikaže učencu namig (besedilo v pravokotniku, primer orodja za namige prikazuje slika 4.14). S pomočjo orodja za namige se učencem dodatno razložijo pravila oziroma podatki o kvizu.

Ob pritisku na gumb `Začni kviz` lahko začnemo z reševanjem kviza. Preden začnemo reševati kviz se nam prikaže opozorilo. Opozorilo nas opozori, da ne smemo zapreti aplikacije med reševanjem kviza (slika 4.19). Prikaz opozorila lahko tudi izključimo.



Slika 4.19: Prikaz opozorila preden začnemo z reševanjem kviza.

Ob pritisku na gumb `Začni kviz` lahko začnemo reševati kviz. V desnem zgornjem kotu zaslona se učencu prikaže časovnik, če gre za časovno omejen kviz. Napisa, ki se še nahajata v zgornjem kotu in si sledita od leve proti desni sta: naslov kviza ter številka vprašanja katerega rešuje učenec. Zavihki `Profil`, `Predmeti` in `Ocene` se onemogočijo. Vprašanje lahko poleg besedila vsebuje še sliko.

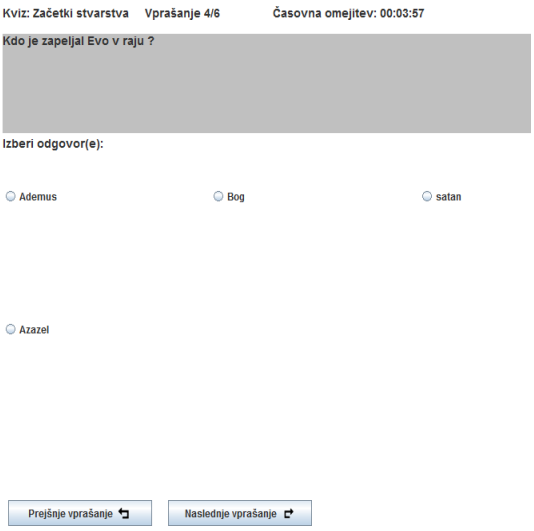
Učenca se ob izteku časa ali pri oddaji vseh odgovorov, preusmeri na zaslon, ki prikazuje uspešnost kviza (slika 4.21). Učenec se lahko pomika naprej in nazaj med posameznimi vprašanji (slika 4.20). Pred oddajo nerešenih vprašanj, se učenca opozori. V dialogu se učenca vpraša, ali želi nadaljevati reševanje kviza. Učenec lahko nadaljuje z reševanje kviza ali pa odda vse rešene odgovore in zaključi kviz. Ko učenec uspešno konča kviz se mu prikaže zaslon, ki prikazuje uspešnost kviza. V levem spodnjem kotu zaslona se nahajajo gumbi z napisom od leve proti desni: `Rešitve kviza`, `Ponovi kviz`, `Naslednji kviz` in `Nazaj na lekcijo`.

Gumb `Rešitev kviza` je omogočen takrat, ko se izteče obdobje izvajanja kviza ali, ko se učencu iztečejo poskusi reševanja kviza. Ob pritisku na ta gumb, se učenca preusmeri na začetno vprašanje. Odtod naprej lahko preveri pravilnost svojih odgovorov.

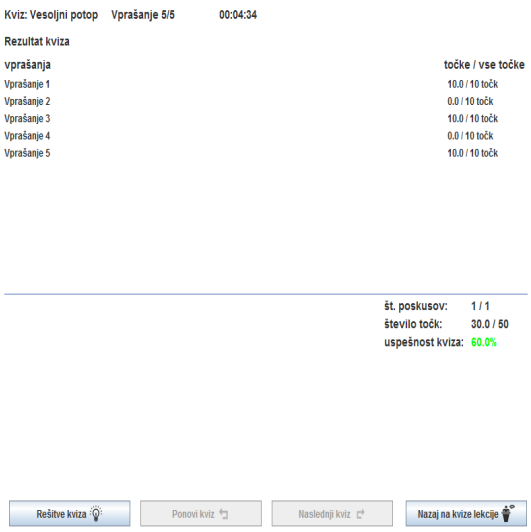
Gumb `Ponovi kviz` je omogočen takrat, ko ima učenec na voljo še dovolj poskusov ali pa gre za kviz z neomejimi poskusi.

Reševanje naslednjega kviza se omogoči takrat, ko ima na voljo še kakšen kviz. V primeru, da je neuspešno rešen osnovni kviz, se lahko učenca pre-

usmeri na reševanje enega ali več podkvizov. Naslednji kviz izbiramo glede na vrsto trenutnega kviza. Začetnemu kvizu sledi osnovni ali testni kviz, osnovnemu kvizu podkviz, osnovni ali testni kviz. Vodeno reševanje kvizov je tako popolnoma odvisno od tega, kako učitelj sestavi kvize.



Slika 4.20: Reševanje kviza.



Slika 4.21: Prikaz uspešnosti kviza.

4.6.3 Zavihek `Ocene`

V zavihku `Ocene` se nahajajo ocene vseh predmetov učenca (slika 4.22). Zaključna ocena se izračuna na podlagi vseh vidnih ocen. Ocene predmetov, ki vsebujejo znak `/`, še niso vidne učencu. Vidne bodo takrat, ko jih učitelji potrdijo. Primer programske kode, ki vrne ocene določenega predmeta:

```
1  private Ocene vrniOcenoUporabnikaDolocenegaPredmeta(int id_pr){
2   ArrayList<Ocene> oc = new ArrayList();
3   PreparedStatement stmt = null;
4   ResultSet rs = null;
5   try {
6     stmt = con.prepareStatement("select o.ocena_predmeta,"
7     + "o.prikazi_oceno "
8     + "from ocene o "
9     + "where o.id_predmet=? and o.Id_uporabnik=?");
10    stmt.setInt(1, id_pr);
11    stmt.setInt(2, trenutniuporabnik.getIduporabnika());
12    rs = stmt.executeQuery();
13    if (rs != null) {
14          while (rs.next()) {
15                int ocena = rs.getInt(1);
16                boolean pri = rs.getBoolean(2);
17                oc.add(new Ocene(ocena, pri));
18          }
19         }
20
21   } catch (Exception e) {
22         System.out.println(e.getMessage());
23   }
24   return oc.isEmpty() ? null : oc.get(0);
25
26 }
```

Funkcija `vrniOcenoUporabnikaDolocenegaPredmeta` zahteva natanko en parameter tipa `int`. Na začetku inicializiramo spremenljivke tipa `ResultSet`, `PreparedStatement` in `ArrayList` <`Ocene`>. V bloku `try/catch` izvedemo funkcijo `prepareStatement`, ki vsebuje kot parameter poizvedbo. Rezultat, ki ga vrne poizvedba se shrani v spremenljivko `rs`. S pomočjo `while` zanke se sprehodimo skozi rezultate, ki jih shranimo v seznam. Na koncu vrnemo spremenljivko tipa `Ocene`. Razred `Ocene` vsebuje v konstruktorju dva parametra, in sicer dejansko oceno (celo število) in prikaz ocene (vrednost `true`

ali `false`). V primeru napake, napako izpišemo in vrnemo spremenljivko tipa `Ocene`, ki ima vrednost `null`.

Slika 4.22: Prikaz ocen učencev.

4.7 Testiranje in analiza aplikacije

Podpoglavje opisuje, kako smo se lotili testiranja aplikacije ter analizo končne rešitve.

4.7.1 Testiranje

Testiranje posameznih delov programske kode je potekalo med razvojem aplikacije. Vse slikovne in `pdf` datoteke se ob zapisu v podatkovno bazo prenesejo v datoteko `serverdp`. V podatkovno bazo pa se shranjujejo samo povezave na datoteke. Datoteka `serverdp` služi za simulacijo strežnika, na katerem se nahajajo datoteke, ki jih uporablja aplikacija. Primer poti nalaganja datotek v mapo `serverdp`, če gre za lekcije:

- `slika - serverdp/obdobje_id/slika/lekcije_id/ime_datoteke`,
- `pdf - serverdp/obdobje_id/pdf/lekcije_id/ime_datoteke`.

Pot datoteke, če gre za vprašanja:

- `slika - serverdp/obdobje_id/slika/vprasanja_id/ime_datoteke`,
- `slike - serverdp/obdobje_id/slike/vprasanja_id/ime_datoteke`.

Posamezne datoteke poiščemo s pomočjo funkcije `poiscidatoteke`.

```
1  private File poiscidatoteke(String tabela,
2  String imedat,int tip, int idobdobje, int primarykey) {
3
4   File vrnjena = null;
5   File trenutne = null;
6   switch (tip) {
7    case 0:
8     trenutne = new File(lokacija + File.separator +
9      "obdobje_"+ idobdobje + File.separator + "slika");
10          break;
11    case 1:
12     trenutne = new File(lokacija + File.separator +
13     "obdobje_"+ idobdobje + File.separator + "pdf");
14          break;
15    case 2:
16     trenutne = new File(lokacija + File.separator +
17     "obdobje_"+ idobdobje + File.separator + "slike");
18          break;
19   }
20
21 trenutne = new File(trenutne.getAbsolutePath()
22                 + File.separator + (tabela + "_" + primarykey));
23
24 for (int i = 0; i < trenutne.listFiles().length; i++) {
25   String pott = trenutne.listFiles()[i].getAbsolutePath().trim();
26    if (pott.contains(imedat.trim())) {
27          vrnjena = new File(pott);
28          break;
29    }
30 }
31 return vrnjena;
32 }
```

Primer iskanja slike, če gre za vprašanje:

```
1 slvp = poiscidatoteke("vprasanja", trvp.slika,
2 0, idobdobjavprasanja(trvp.id_kviz, 0), trvp.id_vprasanja);
```

Pomen vrednosti parametrov (od prvega naprej):

- gre za vprašanje (lahko tudi niz `lekcije`, če gre za lekcijo,
- ime datoteke,
- vrsta datoteke (0 - slika, 1 - `pdf` datoteka, 2 - slike),

- `id` obdobja,
- `id` vprašanja (lahko tudi `id` lekcije, če gre za lekcijo.)

Popravljanje in iskanje napak (semantičnih in sintaktičnih) v programski kodi nam je zelo olajšal razhroščevalnik v programskem okolju Netbeans [7]. Netbeans dovoljuje postavljanje zaustavitvenih točk in pregled nad delovanjem posameznih niti. Zaustavitvene točke povejo programu, od kje naprej naj začne shranjevati vrednosti programskih spremenljivk. Pri testiranju aplikacije smo ustvarili množico različnih tipov uporabnikov. Na ta način smo lahko testirali vse funkcionalnosti, ki jih ima na voljo določen uporabnik.

4.7.2 Analiza

Končna programska rešitev izpolnjuje vse zahteve in funkcionalnosti, ki smo si jih zadali pred izdelavo aplikacije (vse zahteve in funckionalnost so podrobno opisane v poglavju 3). Fleksibilen videz aplikacije omogoča, da se prilagaja različnim velikostim zaslonov. Pri razvoju aplikacije nismo imeli večjih težav. Pojavljale pa so se vedno nove ideje o dodatnih funkcionalnostih, ki bi še dodatno izboljšale izgled grafičnega vmesnika in uporabniško izkušnjo (npr. prostor za obvestila). Zaradi njihove obsežnosti in časa, ki ga bi potrebovali za implementacijo, se nismo odločili za njih. Sama izdelava aplikacije je zanima tudi s strani številk:

- 26188 vrstic programske kode,
- 56 sličic, ki so se uporabile kot ikone (npr. za gumbe),
- 35 razredov,
- več kot 559 funkcij,
- 197 uvozov programskih knjižnjic,
- 24 jar datotek,

- velikost projekta je 12,6 MB,
- velikost podatkovne baze je 7,6 MB,
- 9 shranjenih procedur,
- velikost končne rešitve (jar datoteke) je 1,41 MB.

Čas, ki smo ga porabili za programiranje, izdelavo in oblikovanje sličic znaša skupaj približno 2 meseca. Pri razvoju večjega sistema je potrebno večje število razvijalcev, saj tako pohitrimo razvoj sistema.

Poglavje 5

Sklepne ugotovitve

Končni izdelek diplomske naloge je delujoč prototip enostavnega učnega sistema. Grafični uporabniški vmesnik je fleksibilen, enostaven za uporabo in zajema bistvene funkcionalnosti uporabniških skupin.

Učitelji lahko sestavljajo učno snov in kvize, pridobivajo ocene s pomočjo kvizov in spremljajo statistične podatke o uspešnosti svojih učencev. Skrbniki imajo pregled nad uporabniki, jih potrjujejo ter dodajo učiteljem po potrebi nove predmete. Učenci pridobijo znanje s pomočjo lekcij in kvizov. Kvizi so sestavljeni tako, da učenec napreduje na naslednji kviz samo v primeru uspešno rešenega trenutnega kviza. V primeru, da učenec doseže slabši rezultat (manj kot 50%) le tega izboljša z reševanjem enega ali več podkvizov. Učenec tako utrdi potrebno znanje za ponoven preizkus na kvizu, ki ga prvotno ni prestal.

Postavitev vizualnih gradnikov na zaslon grafičnega vmesnika in razširitve `swing` komponent so nam vzele kar nekaj časa, saj nismo uporabili vgrajega orodja, ki ga ponuja Netbeans. Programsko kodo za izgradnjo in razširitev posameznega objekta (npr. tabele) smo pisali direktno v kodo. Ta način smo izbrali zato, ker uporaba Netbeans `GUI builder-ja`[9] ne nudi enostavnega načina za razširitev `swing` komponent. Nudi `grouplayout`[10], ki za našo implementacijo grafičnega vmesnika ne bi prišel v poštev, saj je potrebno zamenjevati vsebovalnik z ustreznimi panelami. Čas izdelave končne rešitve

bi zelo pohitrili, če bi Netbeans omogočil razširitve posameznih komponent direktno v kodi.

V času izdelave diplomske naloge smo zelo poglobili svoje poznavanje Jave in poizvedovalnega jezika SQL. Poleg tega pa smo se naučili uporabljati bližnjice (npr. skrivanje in razkrivanje funkcij, itd.) v programskem okolju Netbeans [23], ki so nam olajšale pisanje programske kode.

5.0.3 Možnosti za nadaljnje delo

Protip aplikacije je namenjen manjših izobraževalnim ustanovam in posameznikom, ki učijo manjšo skupino ljudi. S pomočjo aplikacije lahko utrdijo znanje svojih učencev, pridobivajo ocene in se posložujejo inovativnih načinov učenja (uporaba kvizov).

Možnosti za nadaljnje delo je veliko. Na samem začetku je potrebno postaviti podatkovno bazo na strežniku in spremeniti povezovalne parametre. Pred predajo aplikacije izobraževalnim ustanovam ali drugim uporabnikom je potrebno aplikacijo testirati še z vidika varnosti in kapacitet, ki jih premore podatkovna baza. Vizualni videz grafičnega vmesnika se lahko izboljša in prilagodi posameznim zahtevam uporabnikov. Sistem se lahko prevede v druge jezike. Kljub vsem možnostim za nadaljnje delo predstavlja prototip učnega sistema zelo dobro podlago za izdelavo resnejšega učnega sistema.

Literatura

[1] Adobe Photoshop. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Adobe_Photoshop. [Dostopano 18. 10. 2015].

[2] Adobe Illustrator. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Adobe_Illustrator. [Dostopano 18. 10. 2015].

[3] ATutor. [Online]. Dosegljivo: http://www.atutor.ca/. [Dostopano 18. 10. 2015].

[4] Blackboard Learn. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Blackboard_Learn. [Dostopano 18. 10. 2015].

[5] Blackboard 9.1 Known Issues. [Online]. Dosegljivo: http://www.fresnostate.edu/academics/blackboard/knownissues-91.html. [Dostopano 18. 10. 2015].

[6] C++. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/C%2B%2B. [Dostopano 18. 10. 2015].

[7] Debugger and Profiler. [Online]. Dosegljivo: https://netbeans.org/features/java/debugger.html. [Dostopano 20. 10. 2015].

[8] ICEpdf Overview. [Online]. Dosegljivo: http://www.icesoft.org/java/projects/ICEpdf/overview.jsf. [Dostopano 20. 10. 2015].

[9] GUI Builder. [Online]. Dosegljivo: https://netbeans.org/kb/docs/java/quickstart-gui.html. [Dostopano 23. 10. 2015].

[10] GroupLayout. [Online]. Dosegljivo: https://docs.oracle.com/javase/7/docs/api/javax/swing/GroupLayout.html. [Dostopano 23. 10. 2015].

[11] Idrijska čipka. [Online]. Dosegljivo: https://sl.wikipedia.org/wiki/Idrijska_%C4%8Dipka. [Dostopano 20. 10. 2015].

[12] Java. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Java_(programming_language). [Dostopano 18. 10. 2015].

[13] JFreeChart. [Online]. Dosegljivo: http://www.jfree.org/jfreechart/index.html. [Dostopano 20. 10. 2015].

[14] Jtree. [Online]. Dosegljivo: https://docs.oracle.com/javase/tutorial/uiswing/components/tree.html. [Dostopano 20. 10. 2015].

[15] Brandl, Klaus. "Are you ready to "Moodle"."Language Learning & Technology 9.2 (2005)

[16] Learning management system. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Learning_management_system. [Dostopano 28. 10. 2015].

[17] Microsoft JDBC Driver for SQL Server. [Online]. Dosegljivo: https://msdn.microsoft.com/en-us/sqlserver/aa937724.aspx. [Dostopano 20. 10. 2015].

[18] Moodle. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Moodle. [Dostopano 18. 10. 2015].

[19] Moodle general help. [Online]. Dosegljivo: https://moodle.org/mod/forum/view.php?f=32&page=0. [Dostopano 18. 10. 2015].

[20] Moodle new features. [Online]. Dosegljivo: https://docs.moodle.org/29/en/New_features. [Dostopano 18. 10. 2015].

[21] Moodle statistics. [Online]. Dosegljivo: https://moodle.net/stats/. [Dostopano 18. 10. 2015].

[22] Notepad++ About. [Online]. Dosegljivo: https://notepad-plus-plus.org/. [Dostopano 18. 10. 2015].

[23] Netbeans IDE. [Online]. Dosegljivo: https://netbeans.org/. [Dostopano 18. 10. 2015].

[24] Rainbow table. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Rainbow_table. [Dostopano 18. 10. 2015].

[25] SQL Server Language Reference, Set nocount. [Online]. Dosegljivo: https://msdn.microsoft.com/en-us/library/ms189837.aspx. [Dostopano 20. 10. 2015].

[26] SwingWorker. [Online]. Dosegljivo: https://docs.oracle.com/javase/7/docs/api/javax/swing/SwingWorker.html. [Dostopano 20. 10. 2015].

[27] SQL Server 2014 Management Studio. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/SQL_Server_Management_Studio. [Dostopano 18. 10. 2015].

[28] SQL Server. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Microsoft_SQL_Server. [Dostopano 18. 10. 2015].

[29] T-SQL. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Transact-SQL. [Dostopano 18. 10. 2015].